Advancing Digital Storage Innovation

EIOW – exa-scale IO workgroup

Status Update Q3, 2012

Participants

- University of Paderborn
- University of Mainz
- Barcelona Supercomputing (BSC)
- DDN
- Fujitsu
- TU Dresden
- University of Tsukuba
- Hamburg University
- TACC
- NCSA
- HDF group
- MPG/RZG
- Juelich
- Goethe Universitat Frankfurt
- ZIH
- DKRZ
- Netapp
- Tokyo Institute of Technology
- Micron
- Xyratex
- DSSD
- Sandia
- PNNL
- Cray
- DOE
- PSC
- LRZ
- HLRS
- CEA
- T-Platforms
- Partec-
- EOFS
- STFC

Contents

- Mission
- Problem space
- Process followed
- Requirements overview: Munich, Portland, Tokyo
- Architecture decomposition
- Future plans

The EIOW mission

Purpose of EIOW

- Let HPC application experts explain requirements for next generation storage
- Architect, design, implement an open source set of exa-scale I/O middleware
- Approach
  - Gather requirements: Europe 02/12, US 04/12, Japan 05/12
  - Design the architecture – breakout groups Barcelona 09/12
  - Analyze architecture for completeness
  - Begin implementation
- So far around 35 participating organizations

Some other goals

- EIOW is an open effort
  - EOFS supported workgroup since inception
  - Everything is being published on the web
    - And actively being copied and amended
  - We will move in the direction of IETF style controlled openness

- EIOW intends to be a ubiquitous middleware
  - An agreed, eventually standardized API for applications / management is targeted to be uniform
  - We hope to be an implementation of choice for researchers to study, amend, influence and change
    - Such research projects have started to spring up
  - A storage access API allowing storage vendors to bolt it onto their favorite data object and metadata stores

Exa scale clusters

- Exa scale systems
  - 10^8 cores – each ~10GF/sec, each ~1G RAM
  - 1,000 cores / node, 0.5 TB RAM / node (10 TF / node)
  - 100K cluster nodes, 50 PB RAM / cluster
  - Mem BW 5 TB/sec - System network BW 200 GB/sec
  - I/O: 300 TB / sec, one node 3 GB / sec
  - File system > 1 EB
  - I/O nodes likely not more than 1,000 – 300 GB/sec

- Technology revolutions
  - File system clients will have ~10,000 cores
  - Architectures will be heterogeneous
  - Flash and/or PCM storage leads to tiered storage
  - Anti revolution – disks will only be a bit faster than today

Exa-scale GAP

- Again, looking at Trinity (and at presentations by myself from 2010) – there is a big 10x gap between the requirements for 2016 (100 PF's) and 2018 (exa-scale)

- From high level
  - Current methods break down but performance increase of flash over disk allows one to plough on from 10PF to 100PF

  - What is in the cards for 100PF - > 1EF.
    - Network bandwidths of around 200 GB/sec
    - Matching "cache" storage bandwidth
  - Can storage local to every node (to increase aggregate BW) be avoided?

What is the problem space?

Some examples what you cannot do with file systems

- Overlapping stripes from different clients
  - Very costly to write
  - An I/O library can detect this
- I/O models, free of locking, synchronization barriers
  - Very similar to what HPC applications do anyway
  - Tuned to HPC like Hadoop was to map-reduce
- Compiler supported speculative & aligned read-ahead
- Allow applications to simply map data structures
- PGAS like location / layout descriptors
- Integrate HDF5 / NetCDF formats into file system
  - Do not use file I/O do to metadata (current practice)

Process

Process

- Standard software architecture processes from Software Engineering Institute (SEI) at CMU

- Quality Attribute Workshop (QAW)
  - For requirements gathering

- Attribute driven design
  - To build an architecture

- Architecture Tradeoff & Analysis Method (ATAM)
  - Is architecture suitable for its purpose?

- Active Reviews for Intermediate Designs (ARID)
  - Is architecture sufficient to start implementation?

Quality Attribute Scenarios

- The entire methodology centers around use cases
- They are called Quality Attribute Scenarios

Scenario Refinement

Table 2: Blank Scenario Refinement Table

Scenario Refinement for Scenario N		
Scenario(s):		
Business Goals:		
Relevant Quality Attributes:		
Scenario Components	Stimulus:	
	Stimulus Source:	
	Environment:	
	Artifact (If Known):	
	Response:	
	Response Measure:	
Questions:		
Issues:		

Requirements vs. Architecture Analysis

- Broadly similar processes
- In the analysis it is mandatory that ***all*** relevant qualities are addressed
- Qualities – typically a list of 5-10
  - Performance
  - Modifiability
  - Testability
  - Usability – including diagnose-ability
  - Availability
  - Security

xyratex

Conclusions from 1st workshop (Munich 02/12)

Requirements gathering

- Some 40 requirements were gathered
- Usual suspects were present
  - Performance, scalability, diagnostics
- After voting some of the top requirements were
  - Behavior indicators
  - Integrity
  - Data properties
  - Stackable interfaces
- All of these point in the direction of *guided interfaces*

Examples of guided mechanisms

- Applications indicate near future re-use of data
  - Avoid movement, keep in cache or memory
- Indicate suitable stripe width
- Specify acceptable levels of integrity
- Assist the storage system with read-ahead decisions
- Indicate when transactions need to become persistent

Interface examples

- Build layers of metadata
  - Retain transactional interfaces
  - Retain data integrity across layers
  - Flexible definitions of schemas, both for data & metadata
  - Keep it simple
  - What sort of container comes with a file / job / ….?
- Many people asked me "how can you keep this simple?"
  - Tentative analogy "SQL made most database apps simple."
- Life cycle indicators
  - Prefetching / staging, cache flushing / retention, read-ahead
  - What sort of work database is this?

Guided vs. Automatic

- Example
  - Do not flush checkpoints to disk, keep last two in “cache” layer
  - Not automated cache management decision, it’s guided
- But … what automatic mechanism helps if the cache is full?
- Automatic over-rides still necessary
  - Can these be computer learning based?
- Question –
  - What are the automatic overrides?
  - How can we avoid pitfalls we intended to eliminate?

Developments since 2nd workshop (Portland 4/14)

Summary of the meeting

- Was more difficult – everyone had to get up to speed

- After the meeting a FastForward proposal was prepared
  - Pittsburgh Supercomputing Center PSC (prime)
  - NCSA
  - TACC
  - Xyratex
  - HDF5group

- Comments
  - The technical part will be posted on EIOW org

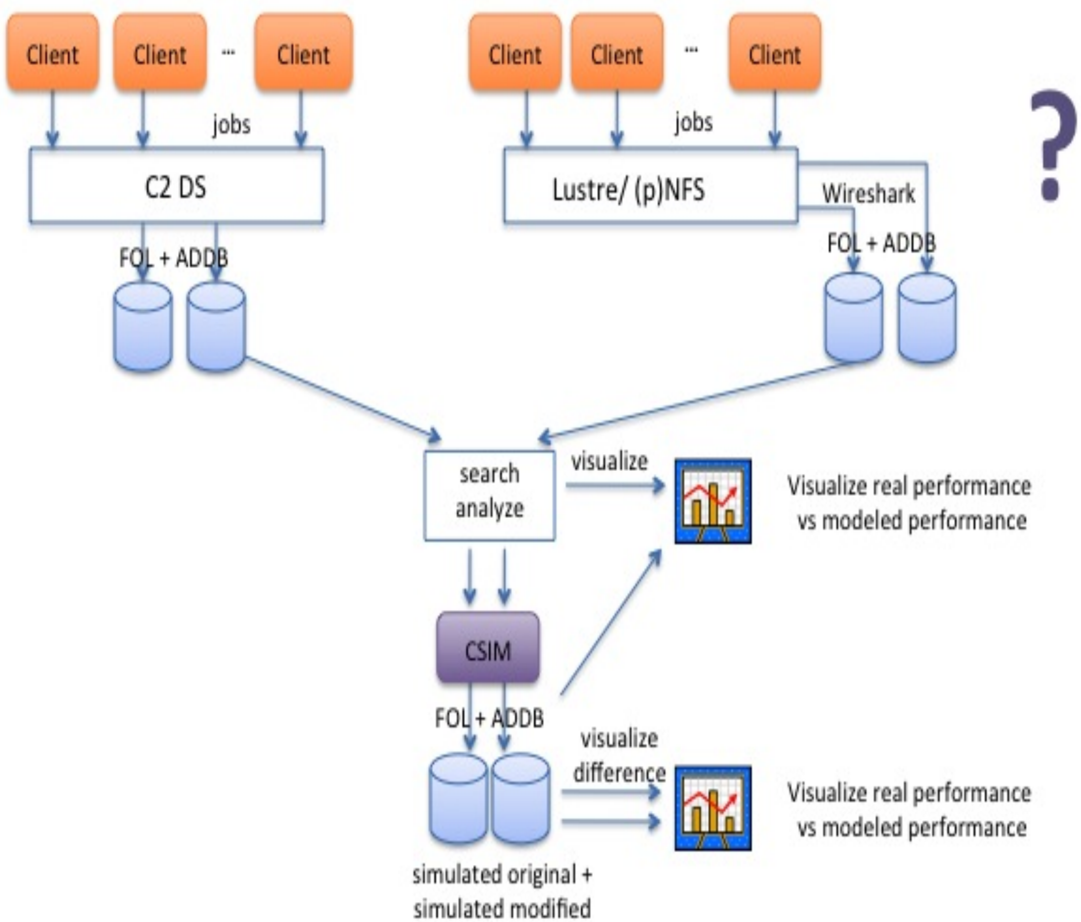
Technical issues from 2nd meeting

- Data commit model
  - Applications will indicate “sync point”
  - Move between snapshots
  - Transactional model unexplored
    - these could be long running transactions, but the “heat is off”
- PGAS redundancy & storage
  - Can a network raided – memory map be used
  - Coarse grained transactional memory update
- Network monitoring, diagnosis, modeling was discussed

Questions that came up during the meeting

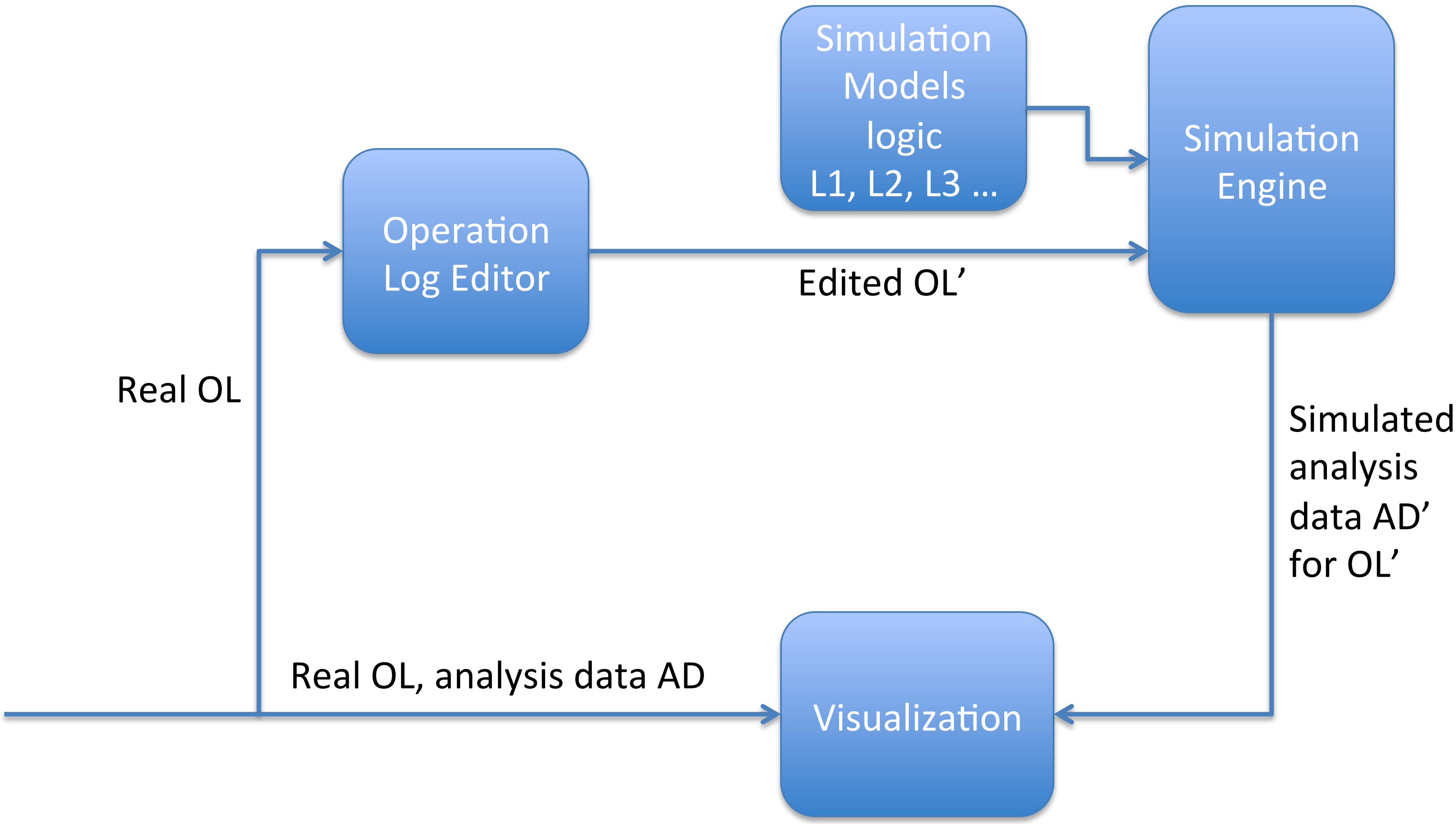
- Namespace of containers
  - Want to avoid POSIX namespace
  - Likely a flat namespace
- Namespace inside containers
  - Many: POSIX (RO?), HDF5, NetCDF, Hadoop
- Distributed containers - misc
  - Schema's for containers may reference / include other schemas
  - Transactions act on one container
- Discussion on reading data (vs generating data)
  - 1 exa-byte / day example from some telescope
  - Possibly the group said "out of scope"

Diagnostics / map reduce / visualization



- ADDB views in XRTX-ClusterStor Mgmt tool:
  - monitoring
  - predictive failure analysis
  - query language, drilling
  - post-factum analysis
- simulation:
  - feed fol and addb traces to a simulator
  - how system behaviour would change with a different number of disks?
  - how mpi job would interfere with a backup process?
- Plugins:
  - asynchronous publish-subscribe consumption of matching FOL records
  - replication, migration, backup, HSM, indexing, fsck

Another view



Requirements meeting 3 (Tokyo 5/12)

Interfaces & attributes of distributed containers

- File-system compartmentalization
- Groups data / metadata objects for job families
- Schema definition—for contents:
  - what data / MD objects are present
  - what structure inside them—a la Hercule
  - dependencies between objects
  - what ILM / SLA attributes for the objects
  - what actors (threads, clients) access the objects
- Storage interface: how are DC stored:
  - Native—distributed DB & object store
  - In file-system—a directory tree (cf OS X Bundles)
- Access interface: how application use DC:
  - Distributed middleware access interface
  - Local interface—leveraging FS underneath

Availability

- Non blocking availability.
- When a device or node fails, don't wait until it is repaired.
- Change file layouts to direct writes to a different set of nodes.
- Writes are never degraded.
- Reads are in degraded mode until the failure is repaired.
- SNS parity de-clustered repair utilises the total pool throughput: very fast (minutes).
- Originally proposed independently by us & Lee Ward (Sandia).

Nested / layered constructions

- Nested constructions
  - E.g. implement HDF5 inside
    - A storage system supported metadata database
    - Storage system data objects

- Legacy applications need a file system view
  - Undecided if / how to address this

- Question – how to specify
  - What is visible to the legacy system
  - What can the EIOW middleware see

Important case study

- Climate modeling
- Generates many large files
  - In Asia usually not in NetCDF but in another format
- Partially unsolved problems
  - How to index, e.g. to find cyclones, max temperatures
  - Is challenging even from DB perspective (nodb was mentioned)
- Clearly an offloaded changelog driven FDMI solution wouldn’t harm
  - See Braam’s Xyratex blog post

FDMI Interfaces for data management

- Operation Log (OL)
  - usable for recovery,
  - usable for data management,
  - usable through the stack,
  - usable by users (subject to access constraints);
- FDML
  - changelog, with subscription tracking last read
  - asynchronous and synchronous plugins:
    - migration, replication, HSM, backup
    - full-text indexing
    - fsck;
  - Update streams – transactional, rich storage operations

Component Decomposition Discussion (9/12 Barcelona)

DC schema examples

1. Directory of files / databases for use by job
2. Format of the files—record sizes
3. Which objects are read-only
4. Data-flow description (utilised by NRS and placement mechanisms)
   a. map-reduce
   b. access patterns
5. SLA: quota, bandwidth, latency, re-use
6. What is the map to a sequential file?

Example use: implement HDF5, NetCDF, HDFS like data schemas

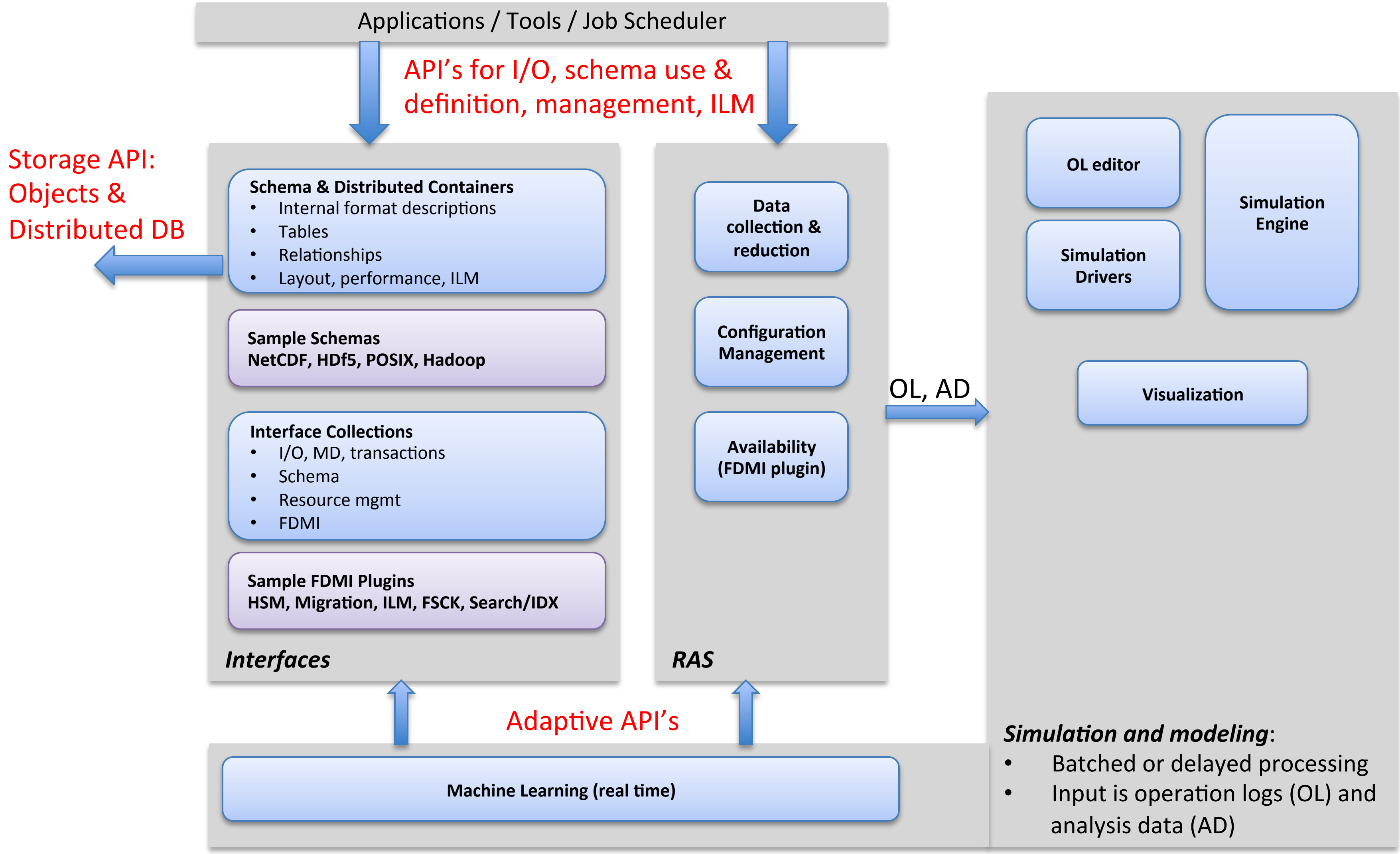
Interfaces: user definable schemas

- behaviour,
- meta-data:
- relations between different data sets
  - multiple data sets related to the same study that must be retained together
- schemas at different levels
  - within files as a storage dictionary (hdf5)
  - among data sets
- annotations,
- layering,
- provenance,
- data and meta-data layouts, redundancy

Metadata: behaviour indicators

- Long(-ish) term behaviour hints:
  - read-ahead, including cross-file
  - data dependency
  - job level behaviour:
    - pre-staging,
    - network request scheduler,
    - phase change (computation vs. IO), initiate checkpoint,
    - fpp vs. shared file,
    - support for workflow applications. E.g., avoid data movement for post-processing.
- Schema and interface components for
  - schema defined attributes of data
  - attributes of DC’s
  - work-flow data base

Component Decomposition



xyratex

Future work

Possibly structure of future work

- Two components
  - core system (schemas, interfaces, HA)
    - probably best prototyped by Industry
  - Adaptive API's, RAS, simulation
    - Probably more suited to Academia
- Evaluate the ideas with prototypes
  - Collaboration between industry and some computing centres
- Community manager
- Next meetings: BOF and ½ day workshop at SC 2012